

Verfahren zur Aufstellung formaler Metamodelle



Daniel Gebauer, Barbara Scherrer, Markus Kühl, Klaus D. Müller-Glaser
Forschungszentrum Informatik FZI
Elektronische Systeme und Mikrosysteme (ESM)
Haid-und-Neu-Str. 10-14
76131 Karlsruhe, Germany
mailto:esm@fzi.de, http://www.fzi.de/esm
phone / fax: +49-721-9654-151 / -159

Abstract

Es wird ein praxisbezogenes und erprobtes Verfahren zur Aufstellung formaler Metamodelle vorgestellt. Die erstellten Metamodelle werden zur Modellierung komplexer Datenstrukturen in modellbasierten Softwaresystemen eingesetzt, wobei die realen Daten Instanzen des entsprechenden Metamodells sind. Der Schwerpunkt der Vorgehensweise liegt auf der intensiven Kommunikation zwischen den Modellierern der Metamodelle und den Experten des zu modellierenden Wissensgebietes. Durch eine iterative Vorgehensweise werden die zu erstellenden Metamodelle schrittweise verfeinert und überprüft. Als Beschreibungsmittel wird die *Unified Modeling Language* (UML) eingesetzt.

I. Einleitung

Datenstrukturen in Softwaresystemen, ganz gleich ob sie in eingebetteten Systemen oder im Desktop-Bereich zum Einsatz kommen, werden immer komplexer und umfangreicher. Mit traditionellen Entwicklungstechniken sind Größe, Komplexität und Konsistenz der zu verwaltenden Datenstrukturen manuell sehr schwer oder nicht mehr zu handhaben. Durch die Einführung modellbasierter Entwicklungstechniken lassen sich Größe und Komplexität der Datenstrukturen beherrschen. Mit der UML [UML] steht ein verbreiteter Standard zur Verfügung, welcher die Modellierung in grafischer Form ermöglicht. Die erstellten Diagramme dienen gleichzeitig als Dokumentation.

Die Einführung formaler Metamodelle zur Beschreibung der Datenstrukturen ermöglicht vereinfachende Abstraktionen und den Einsatz modellbasierter Techniken für die zu beschreibenden Datenstrukturen. Durch Anwendung geeigneter Technologien [Rei05-1], [Bud03] kann aus einem Metamodell effizient Code generiert werden. Compiler und Laufzeitumgebung stellen dabei die syntaktische Konsistenz der Modelle sicher, die man durch Instanziierung des entsprechenden Metamodells erhält. Üblicherweise binden Codegeneratoren auch eine Persistenzschicht ein. Die Qualität der so erstellten Software hängt direkt von der Qualität des Modells ab, wenn man korrekte Modelltransformationen [Rei05-2] und Codegenerierung voraussetzt.

Die vorliegende Arbeit stellt ein praxisbezogenes und erprobtes Verfahren zur Aufstellung formaler Metamodelle vor. Abschnitt II gibt einen Überblick über die zeitliche Vorgehensweise. Die eingesetzten Beschreibungsmittel werden in Abschnitt III vorgestellt. Ein mögliches Vorgehen zur Analyse des Wissensgebietes wird in Abschnitt IV beschrieben. Abschnitt V erläutert wichtige Aspekte bei der eigentlichen Beschreibung des Wissensgebietes.

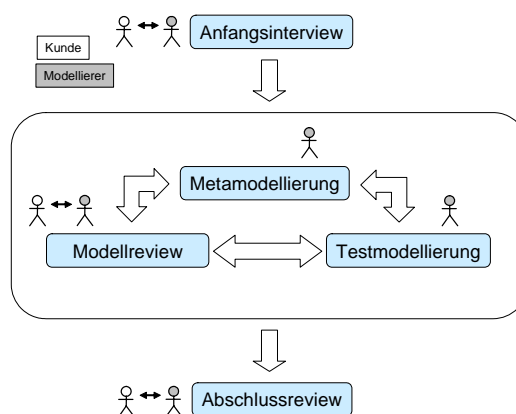


Abbildung 1: Schematische Darstellung der Vorgehensweise

II. Vorgehensweise

Das Verfahren zur Aufstellung formaler Metamodelle ist an agile Methoden von XP (*eXtreme Programming*) angelehnt [Bec00]. In Abbildung 1 ist der Ablauf schematisch dargestellt. Zu Beginn muss das zu modellierende Wissensgebiet analysiert werden. Dazu eignen sich Interviews zwischen den Modellierern und den Experten des Wissensgebietes, die als Kunden bezeichnet werden (siehe Abschnitt IV.A). Danach wird von den Modellierern ein erstes grobes Metamodell erstellt. Dabei wird vor allem auf die Paketstruktur und Klassenhierarchien Wert gelegt. Als Beschreibungsmittel werden ausgewählte Modellierungselemente aus dem in der UML spezifizierten Klassendiagramm eingesetzt. Diese Beschreibungsmittel werden in Abschnitt III vorgestellt.

Zur Überprüfung des Metamodells führen die Modellierer sowohl ausführbare, als auch nicht-ausführbare Testmodellierungen (siehe Abschnitt IV.B) durch. In Modellreviews (siehe Abschnitt IV.C) werden zusammen mit den Kunden die erstellten Metamodelle und eventuell auch die Testmodellierungen überprüft und diskutiert. Diese drei Schritte Metamodellierung, Modellreviews und Testmodellierungen werden iterativ durchlaufen. Dadurch wird das Metamodell immer weiter verfeinert. Diese iterative Vorgehensweise wird durch eine Abnahme des Metamodells in Form eines Abschlussreviews beendet. Das Abschlussreview wird durchgeführt, wenn sich das Metamodell als stabil erwiesen hat und alle gestellten Anforderungen (siehe Abschnitt IV.C) erfüllt. Neben diesen beiden Gesichtspunkten wird während dieses Reviews abschließend überprüft, ob das Metamodell syntaktisch korrekt ist.

III. Beschreibungsmittel

Bevor die Artefakte erläutert werden, die bei der Metamodellierung verwendet werden dürfen, wird das so genannte *Vier-Schichten-Modell* und die *Meta-Object Facility* vorgestellt.

A. Das Vier-Schichten-Modell der OMG

Zur Beschreibung von Abstraktionsebenen im Zusammenhang von Metamodellen eignet sich das *Vier-Schichten-Modell* der Object Management Group [OMG]. Die Einteilung in die vier Ebenen M0 bis M3 wird dabei über den Abstraktionsgrad vorgenommen (siehe Abbildung 2). Im Allgemeinen abstrahiert eine höher gelegene Ebene die darunter liegende Schicht, umgekehrt heißt das, dass die untere Ebene die höher gelegene Schicht spezialisiert.

In der untersten Ebene (M0) werden reale Daten, konkrete Objekte oder Quellcode abgelegt. Die darüber liegende Schicht ist die M1-Ebene für Modelle, wie zum Beispiel konkrete UML- [UML], Simulink- [SIM] oder Statechart-Modelle [Har87]. Diese Modelle beschreiben Artefakte der M0-Ebene. Die Metamodellschicht (M2) beinhaltet die Elemente, mit denen die Modelle der M1-Ebene beschrieben werden.

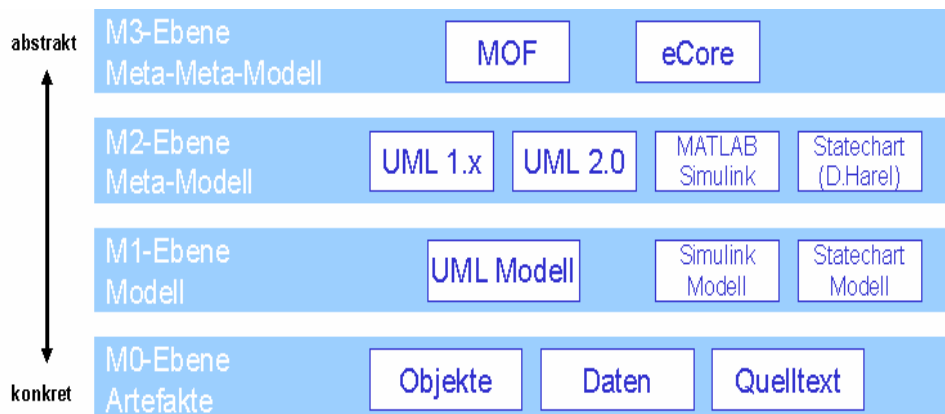


Abbildung 2: Das Vier-Schichten-Modell der OMG

Ein Metamodell spezifiziert somit die abstrakte Syntax eines Modells, das in der konkreten Notation dargestellt wird. Eine abstrakte Syntax beschreibt die Bestandteile einer Sprache und die Zerlegung von Ausdrücken der Sprache in ihre syntaktischen Bestandteile. Diese Beschreibungen sind von der konkreten Notation der Sprache und damit von der konkreten graphischen oder alphanumerischen Darstellung eines Ausdrucks unabhängig [Sch98].

Die oberste Ebene (M3) ist die Metametamodellschicht. Sie besteht aus Metametamodell-Artefakten, die in einem Metamodell verwendet werden dürfen. In der M3-Ebene kommen selbstbeschreibende Metametamodelle wie die *Meta-Object Facility* [MOF] (folgender Abschnitt B) zum Einsatz, so dass keine weiteren Abstraktionsebenen benötigt werden.

B. Die Meta-Object Facility (MOF)

Es gibt mehrere Möglichkeiten, Metamodelle zu beschreiben. Hier wird nur auf MOF-basierte [MOF] Metamodelle eingegangen. Die MOF spezifiziert eine Modellierungssprache, mit der die abstrakte Syntax eines Metamodells modelliert werden kann. Die konkrete Syntax der MOF ist eine Teilmenge der UML [UML], wie sie in der M2-Ebene (siehe Abschnitt III.A) spezifiziert ist. Zur Beschreibung von Metamodellen können somit UML-Klassendiagramme verwendet werden.

C. Zulässige Artefakte bei der Metamodellierung

Bei der Modellierung des Metamodells in UML dürfen ausschließlich die in MOF spezifizierten Beschreibungsmittel verwendet werden, die im Folgenden aufgezählt werden. Die MOF-Konformität kann durch eine geeignete Modelltransformation von der M1-Ebene (UML) in die M2-Ebene (MOF) sichergestellt werden [Rei05-2].

i. Pakete

Pakete werden, wie in der klassischen objektorientierten Softwareentwicklung auch, zur Partitionierung und Gliederung genutzt. Jedes Paket kann zur weiteren Gliederung eine eigene hierarchische Paketstruktur enthalten. Abhängigkeiten zwischen Paketen ergeben sich oft implizit durch ihre jeweiligen Inhalte und deren Beziehungen zueinander. Sie sollten zur besseren Lesbarkeit möglichst explizit durch so genannte *dependencies* im Metamodell dargestellt werden.

ii. Klassen

Klassen sind die wesentlichen Bestandteile eines Metamodells. Sie fassen logische Informationseinheiten zusammen. Die "Daseinsberechtigung" einer Klasse ergibt sich aus der Semantik, die ihr zugeordnet ist.

Über Assoziationen und Generalisierungen können Klassen miteinander in Verbindung stehen. Durch ein solches Zusammenwirken mehrerer Klassen erweitert sich deren jeweilige Semantik. Dabei kann die Semantik einer Klasse, die in mehreren Klassendiagrammen in verschiedene Konzepte (siehe V.B) eingebunden ist, durchaus verschiedene Facetten haben. Es ist daher sehr wichtig, auf eine konsistente Modellierung zu achten.

iii. Attribute

Konkrete Daten der zu beschreibenden Disziplin werden bei der Metamodellierung durch Attribute abgebildet. Dabei muss jedes Attribut einen definierten Datentyp (primitiver Datentyp, String, Enumeration) haben.

iv. Assoziationen

Generell können Klassen über die Assoziationsarten *Assoziation*, *Aggregation* und *Komposition* miteinander in Beziehung stehen. Assoziationen sind die „schwächste“ Verbindungsart. Eine Assoziation zwischen zwei Klassen drückt aus, dass diese Beiden sich „kennen“.

Die Aggregation ist eine Art Mischform zwischen Assoziation und Komposition. Ein Objekt beinhaltet seine aggregierten Objekte, deren Lebenslinien sind aber von der Existenz dieses Objektes unabhängig. Der feine semantische Unterschied zwischen Aggregationen und Assoziationen kann auf Codeebene nicht abgebildet werden. Aggregationen werden daher bei der Metamodellierung nicht verwendet.

Eine Komposition stellt die „stärkste“ Form der Verbindung dar. Sie drückt einen Besitz aus. Dies bedeutet im instanziierten Modell, dass durch Löschen des besitzenden Objektes auch die über Kompositionen verbundenen Objekte gelöscht werden. Daraus folgt, dass ein Objekt nicht mehrere Besitzer haben kann.

Jede modellierte Beziehung wird mit den Eigenschaften Kardinalität, Rollenname, Navigierbarkeit, Assoziationsname und Ordnung ergänzt.

v. *Generalisierungen*

Generalisierungen ermöglichen die Vererbung von Beziehungen und Eigenschaften, sofern deren Sichtbarkeiten nicht auf *private* gesetzt sind. Mehrfachvererbung (d.h. Generalisierungsbeziehungen zu mehreren Klassen) ist prinzipiell erlaubt. Sie dient vor allem dazu, ganze Konzepte auf erbende Klassen zu addieren.

vi. *Kommentare*

Kommentare sind Texte, die bestimmten Modellelementen zugeordnet oder in einem Diagramm frei platziert werden können. Sie dienen zur Dokumentation z.B. der Erläuterung eines Konzeptes.

vii. *Klassendiagramme*

Klassen und ihre Beziehungen werden in Klassendiagrammen modelliert. Sie dienen daher als eigentliche semantische Beschreibung und Dokumentation eines Metamodells. Im Metamodell sollte es ein Vererbungsdiagramm geben, das alle Generalisierungsbeziehungen zwischen den Metaklassen darstellt (siehe Abschnitt V.C). In einem weiteren Klassendiagramm sollten alle Kompositionen gezeigt werden, um den Modellbaum aufzuspannen (siehe Abschnitt V.D). In jedem Paket sollte ein Übersichtsdiagramm existieren, das den Inhalt des Pakets aufzeigt.

IV. Analyse des Wissensgebietes

In diesem Abschnitt werden die einzelnen Schritte der Vorgehensweise beschrieben, die in Abbildung 1 dargestellt sind und zur Analyse des Wissensgebietes eingesetzt werden. Das Abschlussreview wird nicht gesondert erörtert, da es sich dabei um ein Modellreview handelt, bei dem der Schwerpunkt auf der Abnahme des Metamodells liegt.

A. *Anfangsinterview*

Für eine erste Analyse des zu modellierenden Wissensgebietes eignen sich Interviews, sie stehen daher am Anfang der Metamodellierung. In diesem Abschnitt wird die Vorbereitung und Durchführung eines solchen Interviews erläutert. Hierbei werden die Experten des Wissensgebietes als *Kunden* bezeichnet, die Verantwortlichen für die Metamodellierung als *Modellierer*. Ziel dieser Interviews ist es, den Modellierern die Anforderungen an das zu erstellende Metamodell zu vermitteln und erste Strukturen festzulegen.

i. *Vorbereitung*

Vor dem Interview macht sich der Kunde Gedanken über vorhandene Daten aller Art (z.B. Listen, Dokumente, Spezifikationen, Dateien), Strukturen und Prozesse. Dabei sollten vor allem folgende Fragestellungen beachtet werden:

- Welche Informationen müssen und welche Informationen sollen im Modell enthalten sein? Welche Informationen sollen nicht oder dürfen nicht enthalten sein? Wie sollen die Daten weiterverarbeitet werden? Welche Informationen sind dazu zwingend erforderlich? Diese Fragen sind zur Festlegung des Umfangs des zu erstellenden Metamodells essenziell.
- Gibt es bereits strukturierte Daten wie z.B. Tabellen, Dateien oder Programme? Welche erwiesen sich in der Vergangenheit als nützlich? Welche als überflüssig? Wo haben Daten gefehlt? Bei diesen Fragen geht es darum, welche vorhandenen Strukturen eventuell in das Metamodell übernommen werden können.

Für einen zielgerichteten Ablauf ist es wichtig, dass sich die Interviewteilnehmer im Vorfeld auf einen Zeitplan einigen. Dabei sollte genügend Zeit für die Präsentation der vorhandenen Strukturen und Brainstorming eingeplant werden. Die Dauer eines solchen Interviews richtet sich nach der zu erwartenden Größe und Komplexität des zu erstellenden Metamodells. Für ein kleineres Metamodell sollte mindestens ein halber Tag eingeplant werden. Für größere und komplexere Metamodelle können durchaus auch mehrere Tage erforderlich sein. Es ist sinnvoll, diese in Blöcke von ein bis zwei Tagen aufzuteilen, um die jeweiligen Zwischenergebnisse intern durchgehen zu können und „reifen“ zu lassen.

Die Gruppe sollte sich für die Leitung der Interviews auf einen Moderator einigen. Bestehende Hierarchien und Führungsstrukturen sollten für die Dauer des Interviews keine Rolle spielen. Wichtig ist auch

das Hilfsmittel wie z.B. Flipcharts, Whiteboards oder Kärtchen vereinbart werden, deren Verfügbarkeit abgeklärt sein muss. Sinnvollerweise macht sich der Kunde vor dem Interview mit Syntax und Semantik von UML-Klassendiagrammen vertraut.

ii. Durchführung

Das Interview sollte durchgehend protokolliert werden. Falls kein konventionelles schriftliches Protokoll erstellt wird, sollten auf jeden Fall alle „Gedankenblitze“ und neuen Ideen in geeigneter Form, zum Beispiel mit einem der im vorigen Abschnitt beschriebenen Hilfsmittel, festgehalten werden.

Idealerweise werden erarbeitete Konzepte direkt in UML-Klassendiagramm-Fragmenten dokumentiert. Damit ergibt sich bereits eine grobe, aber nicht verbindliche Einteilung in Klassen. Außerdem sind solche Konzepte in Form eines Klassendiagramms übersichtlich und schnell anpassbar. Erarbeitete Konzepte sollten immer wieder hinterfragt werden. Typische Fragen hierfür sind:

- Welchen Sinn hat diese Klasse, Assoziation, Komposition oder Vererbung? Welche Konzepte stellt sie dar und an welchen Konzepten ist sie beteiligt?
- Können die Informationen in der Realität so strukturiert werden, wie das Konzept es fordert?
- Welche Informationen sind zuviel? Ist der Detaillierungsgrad sinnvoll oder stellt das Konzept überflüssige Informationen dar?
- Welche Informationen fehlen in diesem Kontext?

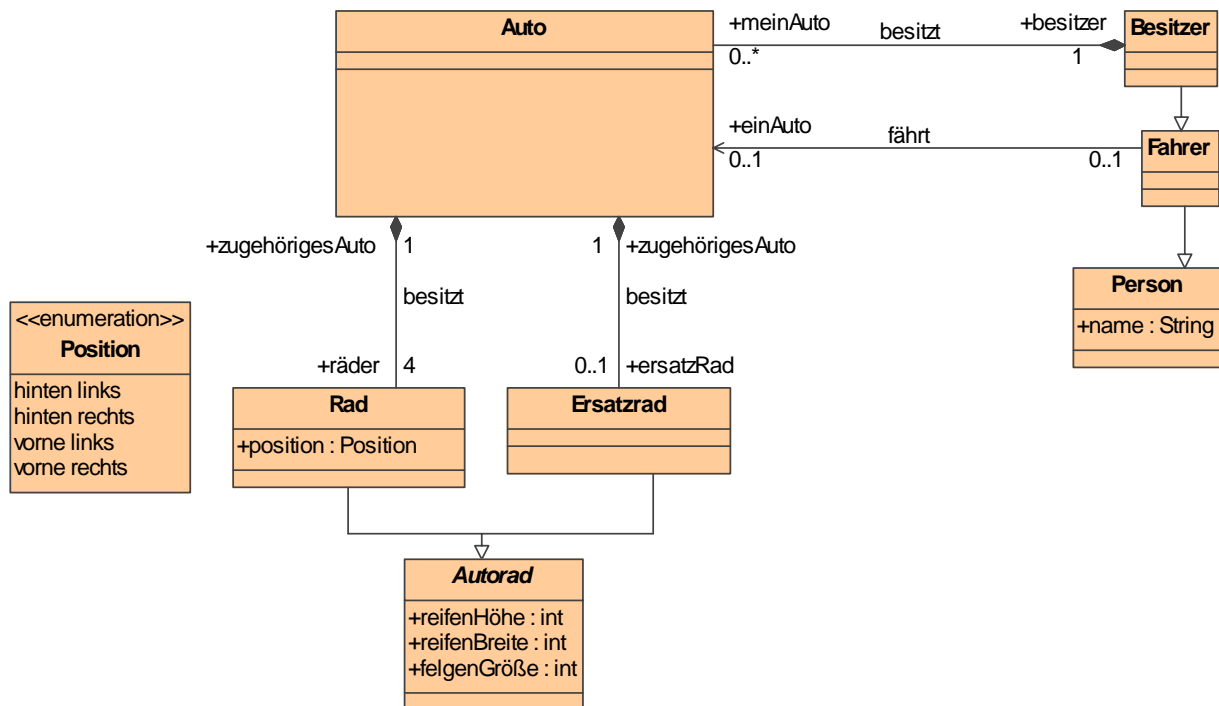


Abbildung 3: Beispielkonzept

Es erweist sich als hilfreich, die Konzepte „vorzulesen“. Die Formulierung „besitzt“ kommt dabei bei Kompositionen und „kennt“ bei Assoziationen zum Einsatz. Diese Technik hilft vor allem bei der Ermittlung der korrekten Kardinalitäten. Das Konzept in Abbildung 3 beschreibt zum Beispiel die Zusammenhänge zwischen einer Person, einem Auto und Rädern. Es kann folgendermaßen vorgelesen werden:

„Ein Objekt der Klasse Auto besitzt genau vier Instanzen von Rad und besitzt höchstens ein Ersatzrad. Jedes Rad hat eine Position mit den möglichen Werten vorne links, vorne rechts, hinten links, hinten rechts. Rad und Ersatzrad sind jeweils ein Autorad und verfügen über die geerbten ganzzahligen Attribute reifenHöhe, reifenBreite und felgenGröße. Ein Besitzer besitzt Autos und kann gleichzeitig als Fahrer und Person agieren. Ein Auto gehört genau einem Besitzer und kennt diesen, nicht aber seinen Fahrer. Beide können übereinstimmen. Ein Fahrer fährt ein oder kein Auto und ist eine Person.“

Klassenattribute sollten erst gegen Ende des Interviews hinzugefügt werden, da es zuerst darauf ankommt, die richtigen Strukturen zu finden. Es können auch Metaklassengruppen gleiche Attribute zugewiesen werden. Dies hilft bei der späteren Erstellung von Vererbungshierarchien. Gegebenenfalls können die Attribute auch erst in den später stattfindenden Modellreviews hinzugefügt werden.

Offensichtliche Vererbungen können während des Interviews erstellt werden, falls dies zur Vereinfachung der Beschreibung dient. Ansonsten gehört die Verfeinerung der Klassenhierarchie nicht zum Umfang des Interviews.

iii. Abschluss

Gegen Ende des Interviews sollten alle Ergebnisse nochmals reflektiert werden. Dazu geben die Modellierer die Anforderungen so wieder, wie sie sie aufgenommen und verstanden haben. Dadurch können Missverständnisse früh erkannt und ausgeräumt werden. Sowohl Kunden als auch Modellierer sollten das entstandene Protokoll gemeinsam durchgehen und verabschieden.

B. Testmodellierung

Das Metamodell sollte über die gesamte Entwicklungszeit hinweg durch Testmodellierungen überprüft werden, um sicher zu stellen, dass die modellierten Konzepte realisierbar sind (korrekte Syntax) und ihren semantischen Zweck erfüllen.

i. Ausführbare Testmodellierung

Ziel der ausführbaren Testmodellierung ist es, Instanzen (= Testmodelle) des Metamodells zu erstellen, welche auf einer Computerplattform ausgeführt werden können. Dieses Vorgehen ist mit den Ideen der *Model Driven Architecture* vergleichbar [MDA].

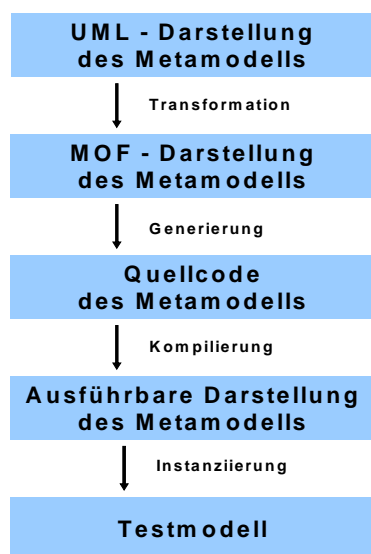


Abbildung 4: Ablauf einer ausführbaren Testmodellierung

Abbildung 4 zeigt den typischen Ablauf einer solchen ausführbaren Testmodellierung. Je nach vorhandener Toolumgebung kann der Ablauf einer ausführbaren Testmodellierung in Einzelheiten vom beschriebenen Vorgehen abweichen. Da das Metamodell mit einem UML-Werkzeug erstellt wurde und daher lediglich eine UML-Darstellung des Metamodells vorliegt, muss diese im ersten Schritt in eine MOF-Darstellung transformiert werden (siehe III.C).

Als nächstes wird aus der MOF-Darstellung mit Hilfe von Codegeneratoren plattformspezifischer Quellcode generiert. Zum Beispiel können die im Metamodell spezifizierten Strukturen gemäß des *Java Metadata Interface* [JMI] nach Java abgebildet werden, wie es in *aquinos.GS* [AQU] realisiert ist.

Im anschließenden Schritt wird dieser Code kompiliert, so dass er in einer ausführbaren Form vorliegt und auf einer kompatiblen Plattform instanziiert und ausgeführt werden kann.

Bei jedem Schritt der ausführbaren Testmodellierung können einige Fehler entdeckt werden. Der erste Schritt (Transformation) schlägt fehl, wenn bei der Modellierung Beschreibungsmittel verwendet wurden, die nicht MOF-konform sind.

Welche Fehler im zweiten Schritt (Generierung) gefunden werden, hängt stark vom eingesetzten Codegenerator ab. Meist können hier nur „handwerkliche“ Fehler lokalisiert werden, die sich nicht direkt auf die Metamodellierung beziehen. Der dritte Schritt (Kompilierung) schlägt fehl, wenn die MOF-Darstellung des Metamodells syntaktische Fehler enthält. Ist der generierte Quellcode nicht kompilierbar, weist das daher direkt auf einen Modellierungsfehler hin.

Im vierten Schritt (Instanziierung) können semantische Fehler entdeckt werden. Dabei können die erstellten Testmodelle entweder manuell oder automatisiert validiert werden. Bei der manuellen Validierung werden Konsistenz und Semantik des Testmodells durch die Eingabe von Daten getestet. Bei Unstimmigkeiten kann das zu überarbeitende Konzept über die betroffenen Typen, welche im Metamodell als Meta-

klassen dargestellt sind, identifiziert werden. Wurden die nötigen Änderungen im Metamodell durchgeführt, muss der komplette Vorgang wiederholt werden.

Da eine manuelle Validierung mitunter sehr zeitaufwändig sein kann, ist es möglich, sie (zumindest in Teilen) durch ausführbare Tests zu automatisieren und somit zu verkürzen. Komplexe Strukturen und Prüfungskriterien, welche ein semantisches Verständnis des Modells voraussetzen, können mit dem derzeitigen Stand der Technik jedoch nur schwer automatisiert werden.

ii. Nicht-ausführbare Testmodellierung

In diesem Fall wird das Metamodell durch Objektdiagramme, welche Instanzen des Metamodells beschreiben, überprüft. Dabei handelt es sich um einen manuellen Prozess, welcher weder die syntaktische noch die semantische Korrektheit des Testmodells vollständig sicherstellen kann. Gerade bei größeren Testmodellen kann es vor allem auch zu Fehlern im Test selbst kommen. Das Verfahren eignet sich aber zur schnellen Überprüfung einzelner Konzepte. Zugleich dokumentieren die erstellten Objektdiagramme die getesteten Konzepte.

C. Modellreview

In regelmäßigen Abständen oder auch, wenn konkrete Änderungswünsche bestehen, sollten Modellreviews durchgeführt werden. Sie sind die konsequente Fortführung der Interviews, die am Anfang der Metamodellerstellung standen. Modellreviews werden in ähnlichem Stil wie die Interviews durchgeführt. Allerdings liegt bei den Modellreviews der Schwerpunkt auf der Validierung des bis dahin entstandenen Metamodells.

i. Vorbereitung

Im Vorfeld müssen Unterlagen eines einheitlichen Standes erstellt und verteilt werden. Alle Teilnehmer sollten Modelle und Unterlagen derselben aktuellen Version vorliegen haben.

Alle offenen Fragen werden in einem Fragenkatalog zusammengestellt. „Baustellen“ bei der Modellierung werden besonders markiert. Für zu komplexe oder schwammige Konzepte sollten Objektdiagramme als Beispielmuster auf der M1-Ebene (siehe Abschnitt III.A) erstellt werden, welche die Schwachstellen verdeutlichen. Dies gilt vor allem auch für Attribute, denen während der Modellreviews im Gegensatz zur Durchführung der Anfangsinterviews besondere Aufmerksamkeit gewidmet wird.

Damit der Ablauf nicht durch ungeeignete oder fehlende Hilfsmittel behindert wird, sollte wieder das Vorhandensein aller gewünschten Ressourcen (Flipcharts, Whiteboards, Kärtchen, etc.) sichergestellt werden.

ii. Durchführung

Die Modellierer stellen das bisher erarbeitete Metamodell vor und belegen die richtige Modellierung mit Beispielmustern, z.B. in Form von Objektdiagrammen oder Testmodellen. Änderungen, welche seit dem letzten Review oder Interview vorgenommen wurden, werden erklärt und begründet. Neben der Terminplanung des nächsten Reviews werden auch die bis dahin fertig zu stellende Arbeitspakete abgesprochen.

Analog zum Anfangsinterview sollte auch ein Modellreview in einer geeigneten Form dokumentiert und protokolliert werden. Änderungen an Konzepten erfolgen idealerweise in Form von UML-Klassendiagrammen. Dabei sind Vorher-Nachher-Ansichten mit entsprechender Dokumentation und Begründung hilfreich.

iii. Abschluss

Wie bei den Anfangsinterviews werden gegen Ende alle Ergebnisse nochmals reflektiert und das Protokoll überprüft. Im Gegensatz zur Durchführung der Anfangsinterviews sollten nun aber hauptsächlich die Kunden die Konzepte so wiedergeben, wie sie sie verstanden haben.

V. Beschreibung des Wissensgebietes

In diesem Abschnitt wird auf verschiedene Themen eingegangen, die bei der Beschreibung des Wissensgebietes durch die Erstellung eines Metamodells eine wichtige Rolle spielen. Dazu gehören die Strukturierung des Metamodells in Pakete, die Konzeptbildung und die Aufstellung der Vererbungs- und Kompositionshierarchien. Ein wichtiger Bereich ist auch der Einsatz von Entwurfsmustern, welche die Modellierung deutlich erleichtern und zur Konsistenz und Übersichtlichkeit des zu erstellenden Metamodells beitragen können.

A. Paketstruktur

Die ermittelten Anforderungen an das Metamodell lassen sich oft auf Antriebe partitionieren. Das heißt, dass sich das Wissensgebiet in Themenbereiche gruppieren und hierarchisieren lässt. Diese hierarchischen „Wissensgruppen“ lassen sich im Metamodell über eine Paketstruktur nachbilden.

Am Anfang können die erstellten Pakete durchaus sich überschneidende Gruppen beinhalten. Es ist jedoch sinnvoll, diese Inkonsistenzen und Redundanzen im Laufe der Metamodellierung aufzulösen. Erweist sich diese Überarbeitung der Gruppen als schwierig, sollte die semantische Einteilung erneut überprüft werden, um die dominierenden Zusammenhänge und Unterschiede heraus zu arbeiten.

Da die gesamte Modellierung eines Metamodells iterativ abläuft (vergleiche Abbildung 1), können sich Hierarchie und Inhalte der Pakete im Lauf der Entwicklung ändern. Am Ende muss ein streng hierarchischer Baum vorliegen, dessen Gruppen untereinander disjunkt sind. Jede Gruppe wird dann durch ein Paket im Metamodell repräsentiert.

B. Konzeptbildung

Ein Konzept ist die Realisierung einer Anforderung im Metamodell. Es wird in einem UML-Klassendiagramm modelliert, wobei die in Abschnitt III.C aufgelisteten Beschreibungsmittel verwendet werden dürfen. Die erstellten Klassendiagramme, die die verschiedenen Konzepte beinhalten, sind daher wichtige Bestandteile der Dokumentation des Metamodells.

Zur Ermittlung von Konzepten bei der Analyse des Wissensgebietes (siehe Abschnitt IV) können gängige Verfahren der agilen Softwareentwicklung angewendet werden. Dazu zählen zum Beispiel *Story Cards*, wie sie bei *eXtreme Programming* eingesetzt werden [Bec00]. Wichtig ist bei der Konzeptmodellierung den richtigen Abstraktionsgrad zu wählen. Er muss den Anforderungen an die Instanzen des Metamodells entsprechen.

Ein Konzept sollte nur dann modelliert werden, wenn es unbedingt benötigt wird und sollte so einfach wie möglich gestaltet sein. Zu beachten ist auch, dass eine Konzeptmodellierung in einem Metamodell nur statische Zusammenhänge abbilden kann, auch wenn semantisch dynamische Aspekte beschrieben werden können. Zum Beispiel lassen sich Statecharts [Har87] durch ein Metamodell abbilden.

Im einfachsten Fall reicht zur Umsetzung eines Konzeptes ein Attribut aus, das einer Metaklasse zugeordnet wird. Meistens wird ein Konzept aber durch mehrere Metaklassen dargestellt, die durch Assoziationen, Kompositionen und Generalisierungen verbunden sind und verschiedene Attribute enthalten. Die Metaklassen eines Konzeptes können auch aus unterschiedlichen Paketen stammen. Komplexe Konzepte, bei denen sehr viele Metaklassen modelliert werden müssen, sollten in kleinere Konzepte aufgeteilt werden. Zusätzlich können diese dann in einem Klassendiagramm als Übersichtsdiagramm zusammengestellt werden.

Generell sollten im Metamodell nur neue Metaklassen angelegt werden, wenn sie zur Erfüllung eines Konzeptes benötigt werden. Metaklassen, die an keinem Konzept teilnehmen, sind überflüssig und werden aus dem Metamodell entfernt. Während der iterativen Entwicklung des Metamodells (vergleiche Abbildung 1) kann sich die Semantik einzelner Konzepte ändern. Diese Änderungen werden in der Modellierung durch die Metaklassen und ihren Beziehungen untereinander eingearbeitet.

C. Vererbungshierarchie

Ein Klassendiagramm im Metamodell sollte die Vererbungshierarchie aller Metaklassen darstellen. Ein solches Diagramm erleichtert vor allem die Überprüfung der Metamodellierung und auch Einarbeitung in ein bestehendes Metamodell, da sich die Generalisierungsbeziehungen zwischen den Metaklassen darin schnell und einfach ablesen lassen.

Um die Vererbungshierarchie zu verfeinern und zu vervollständigen, sollten die vorhandenen Metaklassen in regelmäßigen Abständen auf Gemeinsamkeiten untersucht werden. Dies gilt sowohl für strukturelle Gemeinsamkeiten wie gleiche Assoziationen und Attribute, als auch für semantische Gemeinsamkeiten. Diese ergeben sich, wenn zwei oder mehrere Metaklassen in verschiedenen Konzepten identische oder ähnliche Sachverhalte repräsentieren. Hierbei ist besonders auf eventuell vorhandene Mehrfachvererbung zu achten.

D. Kompositionshierarchie

Die Kompositionshierarchie eines Metamodells ergibt sich aus allen Kompositionsbeziehungen, die zwischen den Metaklassen bestehen. In jedem Metamodell sollte es ein Klassendiagramm geben, das alle diese Kompositionsbeziehungen aufzeigt. Dieses Diagramm ist ebenso wie das Klassendiagramm, das die Vererbungshierarchie darstellt, vor allem bei der Überprüfung der Metamodellierung und bei der Einarbeitung in ein bestehendes Metamodell hilfreich.

Die Kompositionshierarchie eines Metamodells sollte regelmäßig überprüft werden. Dabei ist darauf zu achten, dass die zu einem beliebigen Ausführungszeitpunkt vorhandenen Instanzen der Metaklassen bezüglich ihrer Kompositionsbeziehungen einen streng hierarchischen Baum aufspannen. Dazu müssen alle Objekte des instanziierten Metamodells genau ein „Vaterobjekt“ besitzen, mit dem sie über eine Komposition verbunden sind. Nur der so genannte „Wurzelknoten“ besitzt kein solches Objekt. Von ihm ausgehend lässt sich der gesamte Modellbaum aufspannen. Diese Anforderung gewährleistet, dass über den Wurzelknoten das gesamte Modell erreicht werden kann. Damit wird zum Beispiel ausgeschlossen, dass Objekte verloren gehen. Außerdem wird die Modellorganisation und Übersichtlichkeit verbessert.

E. Entwurfsmuster

Entwurfsmuster abstrahieren von speziellen Modellierungen für wiederkehrende Entwurfsprobleme. Sie stellen verallgemeinerte Lösungen dar, die als eine Art Vorlage bei der Problemlösung immer wieder verwendet werden können. Durch die Veröffentlichung des heute als Standardwerk geltenden Buchs „Design Patterns“ [GoF97] fanden Entwurfsmuster in der Softwareentwicklung eine weite Verbreitung.

Für die Metamodellierung sind prinzipiell alle Strukturmuster interessant, wie zum Beispiel das Kompositions-Entwurfsmuster [GoF97]. Sie sind Lösungsvorlagen für strukturelle Entwurfsprobleme, die bei der Modellierung statischer Zusammenhänge auftreten können. Bei jeder Metamodellierungsaufgabe sollten wiederkehrende Lösungsansätze analysiert und daraus eigene Entwurfsmuster extrahiert und dokumentiert werden. Wie bereits erwähnt, können Entwurfsmuster die Modellierung deutlich erleichtern und zur Konsistenz und Übersichtlichkeit des zu erstellenden Metamodells beitragen.

VI. Zusammenfassung

Das vorgestellte Verfahren zur Aufstellung formaler Metamodelle ist breit einsetzbar und für jede Art der Metamodellierung geeignet. Der Schwerpunkt der Vorgehensweise liegt auf der intensiven Kommunikation zwischen den Modellierern der Metamodelle und den Experten des zu modellierenden Wissensgebietes. Nach einem Anfangsinterview werden die drei Schritte Metamodellierung, Testmodellierung und Modellreview iterativ durchlaufen, bis das gesamte Metamodell in einem Abschlussreview abgenommen wird. In der Praxis hat sich diese iterative Vorgehensweise bewährt, bei der die zu erstellenden Metamodelle schrittweise verfeinert und überprüft werden.

Das hier vorgestellte Verfahren wurde zur Erstellung verschiedener Metamodelle praktisch angewendet. Zum Beispiel wurde im Rahmen des IMMOS-Projektes [IMM] Artefakte der modellbasierten Steuergeräteentwicklung beschrieben. Ein weiteres erstelltes Metamodell mit über 100 Metaklassen bildet heute in ausführbarer Form die Grundlage eines grafischen Modellierungswerkzeuges zur Beschreibung von Netzwerken verteilter, eingebetteter Systeme.

Danksagung

Die vorliegende Arbeit entstand zum großen Teil im Rahmen des Projektes IMMOS [IMM]. Das Akronym IMMOS steht dabei für *Integrierte Methodik zur modellbasierten Steuergeräteentwicklung*. Dieses Forschungsprojekt wird vom Bundesministerium für Bildung und Forschung (BMBF) im Rahmen des Programms "IT-Forschung 2006" gefördert.

Zur Unterstützung wurden die Werkzeuge *aquintos.GS* und *aquintos.M2M* eingesetzt, welche uns freundlicherweise von der Firma *aquintos GmbH* zur Verfügung gestellt wurden [AQU].

Literaturverzeichnis

- | | | | |
|---------|--|-----------|---|
| [AQU] | aquintos GmbH;
http://www.aquintos.com | [OMG] | Object Management Group (OMG);
http://www.omg.org |
| [Bud03] | F. Budinsky, D. Steinberg, E. Merks, R. Ellersick, and T. Grose, „Eclipse Modeling Framework (EMF)“. Addison-Wesley Professional, 2003. | [Rei05-1] | Reichmann, C., Graf, P., Müller-Glaser, K.D., „GeneralStore - A CASE-Tool Integration Platform Enabling Model Level Coupling of Heterogeneous Designs for Embedded Electronic Systems“. 11th IEEE Conference on the Engineering of Computer Based Systems 2004, Brno, Czech Republic, May 2004. |
| [Bec00] | Kent Beck. „eXtreme Programming – das Manifest. Die revolutionäre Methode für Softwareentwicklung in kleinen Teams“. Addison-Wesley, 2000. | [Rei05-2] | Reichmann, C.: Grafisch notierte Modell-zu-Modell-Transformationen für den Entwurf eingebetteter elektronischer Systeme, Dissertation, ITIV, Universität Karlsruhe, 2005. |
| [GoF97] | Erich Gamma et al., „Design Patterns“. Addison-Wesley Professional, 1997. | [SIM] | Simulink® von The MathWorks, Inc
http://www.mathworks.com/products/simulink/ |
| [Har87] | Harel, D., „Statecharts: a visual formalism for complex systems“. <i>Science of Computer Programming</i> , 8 (1987,3), 231-274 | [Sch98] | Hans-Jochen Schneider, „Lexikon der Informatik und Datenverarbeitung“. Oldenbourg, München/Wien 1998. 4.Auflage |
| [IMM] | IMMOS-Projekt;
http://www.immos-projekt.de/ | [UML] | OMG, Unified Modeling Language (UML), Version 2.0;
http://www.omg.org/uml |
| [MDA] | OMG, Model Driven Architecture (MDA);
http://www.omg.org/mda/ | | |
| [MOF] | OMG, Meta Object Facility (MOF), Version 1.4;
http://www.omg.org/mof | | |